Claims

1. A method of analyzing instructions and data to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the method comprising the steps of:

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are multiprocessor unsafe references;

generating a report of the multiprocessor unsafe references; and

modifying the instructions and data based on the report.

2. A method as in claim 1, wherein the instructions and data comprise code that prior to modification is designed for use on a single processor system.

3. A method as in claim 1, wherein the determining steps, the generating step, and the modifying step are repeated until none of the references are determined to be multiprocessor unsafe references, whereby the code is modified to be suitable for use on a multiprocessor system.

4. A method as in claim 1, wherein the instructions and data comprise source code that is analyzed before compilation.

5. A method as in claim 1, wherein the instructions and data comprise object code that is analyzed before being linked to form an executable.

6. A method as in claim 1, wherein the instructions and data comprise object code that is analyzed while being linked to form an executable.

7. A method as in claim 1, wherein the instructions and data comprise interpretable code that is analyzed at run time.

8. A method as in claim 1, wherein the instructions and data are for execution by a virtual machine, and wherein the instructions and data are analyzed by the virtual machine at run time.

9. A method as in claim 1, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

10. A method as in claim 9, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

11. A method as in claim 1, wherein the step of determining which of the references are multiprocessor unsafe further comprises determining which of the references are neither determined to be always multiprocessor safe nor annotated in the code as multiprocessor safe.

12. A method as in claim 11, wherein references can be annotated in the code as multiprocessor safe because the references switch domains at run time.

13. A method as in claim 1, wherein the instructions and data are modified by adding annotations that indicate references outside of their domains are multiprocessor safe.

14. A method as in claim 1, wherein the instructions and data are modified by adding switch domain functions to the instructions and data to change multiprocessor unsafe references outside of a domain into multiprocessor safe references inside the domain.

15. A method as in claim 1, wherein the instructions and data are

modified automatically based on the report of the multiprocessor unsafe references.

16. A method as in claim 1, wherein the instructions and data are modified by an expert system aided by a user.

17. A method as in claim 1, further comprising the steps of:

determining which of the references outside of their domains are purportedly multiprocessor safe references; and

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer.

18. A memory storing information including steps executable by a processor, the steps executable to analyze instructions and data to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the steps comprising:

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are multiprocessor unsafe references;

generating a report of the multiprocessor unsafe references; and

modifying the instructions and data based on the report.

19. A memory as in claim 18, wherein the instructions and data comprise code that prior to modification is designed for use on a single processor system.

20. A memory as in claim 18, wherein the determining steps, the generating step, and the modifying step are repeated until none of the references are determined to be multiprocessor unsafe references, whereby the code is modified to be suitable for use on a multiprocessor system.

21. A memory as in claim 18, wherein the instructions and data comprise source code that is analyzed before compilation.

22. A memory as in claim 18, wherein the instructions and data comprise object code that is analyzed before being linked to form an executable.

23. A memory as in claim 18, wherein the instructions and data comprise object code that is analyzed while being linked to form an executable.

24. A memory as in claim 18, wherein the instructions and data comprise interpretable code that is analyzed at run time.

25. A memory as in claim 18, wherein the instructions and data are for execution by a virtual machine, and wherein the instructions and data are analyzed by the virtual machine at run time.

26. A memory as in claim 18, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a general domain comprising other functions and data referred to by the other functions.

27. A memory as in claim 26, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

28. A memory as in claim 18, wherein the step of determining which of the references are multiprocessor unsafe further comprises determining which of the references are neither determined to be always multiprocessor safe nor annotated in the code as multiprocessor safe.

29. A memory as in claim 28, wherein references can be annotated in the code as multiprocessor safe because the references switch domains at run time.

30. A memory as in claim 18, wherein the instructions and data are modified by adding annotations that indicate references outside of their domains are multiprocessor safe.

31. A memory as in claim 18, wherein the instructions and data are modified by adding switch domain functions to the instructions and data to change multiprocessor unsafe references outside of a domain into multiprocessor safe references inside the domain.

32. A memory as in claim 18, wherein the instructions and data are modified automatically based on the report of the multiprocessor unsafe references.

33. A memory as in claim 18, wherein the instructions and data are modified by an expert system aided by a user.

34. A memory as in claim 18, wherein the steps further comprise the steps of:

determining which of the references outside of their domains are purportedly multiprocessor safe references; and

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer.

35. A memory as in claim 18, wherein the memory is a removable storage medium, fixed disk, RAM or ROM.

36. An analyzer that analyzes instructions and data to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the analyzer comprising:

a reference analyzer that determines which of the instructions and data involve references outside of their domains, and determines which of the references outside of their domains are multiprocessor unsafe references; and

a report generator that generates a report of the multiprocessor unsafe references.

37. An analyzer as in claim 36, further comprising a modifier that modifies the instructions and data based on the report.

38. An analyzer as in claim 37, wherein the instructions and data comprise code that prior to modification is designed for use on a single processor system.

39. An analyzer as in claim 37, wherein the determining steps, the

generating step, and the modifying step are repeated until none of the references are determined to be multiprocessor unsafe references, whereby the code is modified to be suitable for use on a multiprocessor system.

40. An analyzer as in claim 37, wherein the instructions and data comprise source code that is analyzed before compilation.

41. An analyzer as in claim 37, wherein the instructions and data comprise object code that is analyzed before being linked to form an executable.

42. An analyzer as in claim 37, wherein the instructions and data comprise object code that is analyzed while being linked to form an executable.

43. An analyzer as in claim 37, wherein the instructions and data comprise interpretable code that is analyzed at run time.

44. An analyzer as in claim 37, wherein the instructions and data are for execution by a virtual machine, and wherein the instructions and data are analyzed by the virtual machine at run time.

45. An analyzer as in claim 37, wherein the plural domains include a network domain comprising network functions and data referred to by the network

functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

46. An analyzer as in claim 45, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

47. An analyzer as in claim 37, wherein determining which of the references are multiprocessor unsafe further comprises determining which of the references are neither determined to be always multiprocessor safe nor annotated in the code as multiprocessor safe.

48. An analyzer as in claim 47, wherein references can be annotated in the code as multiprocessor safe because the references switch domains at run time.

49. An analyzer as in claim 37, wherein the instructions and data are modified by adding annotations that indicate references outside of their domains are multiprocessor safe.

50. An analyzer as in claim 37, wherein the instructions and data are

modified by adding switch domain functions to the instructions and data to change multiprocessor unsafe references outside of a domain into multiprocessor safe references inside the domain.

51. An analyzer as in claim 37, wherein the instructions and data are modified automatically based on the report of the multiprocessor unsafe references.

52. An analyzer as in claim 37, wherein the instructions and data are modified by an expert system aided by a user.

53. An analyzer as in claim 37, wherein the reference analyzer further determines which of the references outside of their domains are purportedly multiprocessor safe references; and wherein the analyzer further comprises a table generator that generates a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer.

54. A method of dynamically determining where instructions and data result in domain violations when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the method comprising the steps of:

accessing a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

55. A method as in claim 54, wherein the instructions and data comprise interpretable code.

56. A method as in claim 54, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the method.

57. A method as in claim 54, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

58. A method as in claim 57, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve

references to the multiprocessor safe domain are considered to be multiprocessor safe references.

59. A method as in claim 54, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

60. A method as in claim 54, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

61. A method as in claim 60, wherein the instruction or data making the reference is re-executed after being modified.

62. A memory storing information including steps executable by a processor, the steps executable to dynamically determine where instructions and data result in domain violations when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the steps comprising:

accessing a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

63. A memory as in claim 62, wherein the instructions and data comprise interpretable code.

64. A memory as in claim 62, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the steps.

65. A memory as in claim 62, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

66. A memory as in claim 65, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

67. A memory as in claim 62, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

68. A memory as in claim 62, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

69. A memory as in claim 68, wherein the instruction or data making the reference is re-executed after being modified.

70. A checker that dynamically determines where instructions and data result in domain violations when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the checker comprising:

an interface to a table of purportedly microprocessor safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer; and

a reference tracker that tracks references made by the instructions and data; and

a comparator that determines, when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, if the reference is actually to a domain to which that reference is supposed to refer.

71. A checker as in claim 70, wherein the instructions and data comprise interpretable code.

72. A checker as in claim 70, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine includes the checker.

73. A checker as in claim 70, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

74. A checker as in claim 73, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

75. A checker as in claim 70, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data

halts and an error message is generated.

76. A checker as in claim 70, further comprising a modifier that, if the reference is not actually to a domain to which that reference is supposed to refer, modifies the instruction or data making the reference.

77. A checker as in claim 76, wherein the instruction or data making the reference is re-executed after being modified.

78. A checker as in claim 70, wherein the checker is embodied as a function in the instructions and data.

79. A checker as in claim 70, wherein the checker runs concurrently with execution of the instructions and data.

80. A method of analyzing instructions and data and dynamically determining where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the method comprising the steps of:

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are purportedly multiprocessor safe references;

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

81. A method as in claim 80, wherein the instructions and data comprise interpretable code.

82. A method as in claim 80, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the method.

83. A method as in claim 80, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

84. A method as in claim 83, wherein the plural domains further

comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

85. A method as in claim 80, further comprising the steps of:

determining which of the references outside of their domains are multiprocessor unsafe references; and

generating a report of the multiprocessor unsafe references.

86. A method as in claim 85, further comprising the step of modifying the instructions and data based on the report.

87. A method as in claim 80, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

88. A method as in claim 80, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

89. A method as in claim 88, wherein the instruction or data making the reference is re-executed after being modified.

90. A memory storing information including steps executable by a processor, the steps executable to analyze instructions and data and dynamically determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the steps comprising:

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are purportedly multiprocessor safe references;

generating a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and

when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

91. A memory as in claim 90, wherein the instructions and data comprise interpretable code.

92. A memory as in claim 90, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine performs the steps.

93. A memory as in claim 90, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

94. A memory as in claim 93, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe references.

95. A memory as in claim 90, further comprising the steps of:

determining which of the references outside of their domains are multiprocessor unsafe references; and

generating a report of the multiprocessor unsafe references.

96. A memory as in claim 95, further comprising the step of modifying the instructions and data based on the report.

97. A memory as in claim 90, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

98. A memory as in claim 90, wherein if the reference is not actually to a domain to which that reference is supposed to refer, the instruction or data making the reference is modified.

99. A memory as in claim 98, wherein the instruction or data making the reference is re-executed after being modified.

100. A system for analyzing instructions and data and dynamically determining where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the system comprising:

a reference analyzer that determines which of the instructions and data involve references outside of their domains, and determines which of the references outside of their domains are purportedly multiprocessor safe references;

a table generator that generates a table of the purportedly multiprocessor safe references, the table including the domains to which the references are supposed to refer;

a reference tracker that tracks references made by the instructions and data; and

a comparator that determines, when a reference in the table of purportedly microprocessor safe references is encountered during execution of the instructions and data, if the reference is actually to a domain to which that reference is supposed to refer.

101. A system as in claim 100, wherein the instructions and data comprise interpretable code.

102. A system as in claim 100, wherein the instructions and data are for execution by a virtual machine, and wherein the virtual machine implements the system.

103. A system as in claim 100, wherein the plural domains include a network domain comprising network functions and data referred to by the network functions, a storage domain comprising storage functions and data referred to by the storage functions, and a filesystem domain comprising other functions and data referred to by the other functions.

104. A system as in claim 103, wherein the plural domains further comprise a multiprocessor safe domain, and wherein instructions and data that involve references to the multiprocessor safe domain are considered to be multiprocessor safe

references.

105. A system as in claim 100, wherein the reference analyzer further determines which of the references outside of their domains are multiprocessor unsafe references, and wherein the system further comprises a report generator that generates a report of the multiprocessor unsafe references.

106. A system as in claim 105, further comprising a modifier that modifies the instructions and data based on the report.

107. A system as in claim 100, wherein if the reference is not actually to a domain to which that reference is supposed to refer, execution of the instructions and data halts and an error message is generated.

108. A system as in claim 100, further comprising a modifier that, if the reference is not actually to a domain to which that reference is supposed to refer, modifies the instruction or data making the reference.

109. A system as in claim 108, wherein the instruction or data making the reference is re-executed after being modified.

110. A method of analyzing instructions and data to determine where the

instructions and data might result in incorrect results when run on a system having multiple resources of a type used concurrently, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the system configured to use at most one of the resources at a time to execute instructions and to access data from any one domain, the method comprising the steps of:

determining which of the instructions and data involve references outside of their domains;

determining which of the references outside of their domains are unsafe references;

generating a report of the unsafe references; and

modifying the instructions and data based on the report.

111. A method of dynamically determining where instructions and data result in domain violations when run on a system having multiple resources of a type used concurrently, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the system configured to use at most one of the resources at a time to execute instructions and to access data from any one domain, the method comprising the steps of:

accessing a table of purportedly safe references by the instructions and data outside of their domains, the table including the domains to which the references are supposed to refer;

executing the instructions and data; and when a reference in the table of purportedly safe references is encountered during execution of the instructions and data, determining if the reference is actually to a domain to which that reference is supposed to refer.

112. Instructions and data stored in a memory, the instructions and data configured for execution on a multiprocessor system, comprising:

domain definitions whereby the instructions and data are defined as existing in plural domains; and

annotations whereby references outside of their domains in the instructions and data are indicated as being multiprocessor safe.

113. Instructions and data as in claim 112, further comprising instructions to switch domains.

114. Instructions and data as in claim 112, wherein the domain definitions further comprise a makefile.

115. Instructions and data as in claim 114, wherein the makefile comprises a list of domains and a list of files assigned to each domain, whereby instructions and data defined by those files are assigned to the domains.

116. Annotations in instructions and data stored in a memory, the

instructions and data configured for execution on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the annotations comprising reasons why inter-domain references in the instructions and data are multiprocessor safe, whereby analysis of the instructions and data considers the annotated references to be multiprocessor safe.

117. Annotations as in claim 116, wherein the reasons include that the reference is always safe despite it being an inter-domain reference.

118. Annotations as in claim 116, wherein the reasons include that the reference is only used at initialization.

119. Annotations as in claim 116, wherein the reasons include that the reference only can occur when a single processor is executing code.

120. Annotations as in claim 116, wherein the reasons include that the reference is to a constant.

121. Annotations as in claim 116, wherein the reasons include that the reference is to read-only data.

122. Annotations as in claim 116, wherein the reasons include that the reference is of such a nature that interference with another reference is acceptable.

123. Annotations as in claim 116, wherein the reasons include that the reference switches domains so as to be multiprocessor safe.

124. A report stored in a memory, the report resulting from analysis of instructions and data to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the report comprising:

a list of inter-domain references by the instructions and data that the analysis has shown are multiprocessor unsafe; and

for each inter-domain reference, the domains involved in the inter-domain reference.

125. A table stored in a memory, the table resulting from analysis of instructions and data to determine where the instructions and data might result in incorrect results when run on a multiprocessor system, the instructions and data divided into plural domains based on the symbols used to refer to those instructions and data, the

multiprocessor system configured to use at most one processor at a time to execute instructions and to access data from any one domain, the table comprising:

a list of purportedly microprocessor safe references by the instructions and data outside of their domains; and

the domains to which the references are supposed to refer.